(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2004 年4 月29 日 (29.04.2004)

PCT

(10) 国際公開番号
WO 2004/036420 A1

(51) 国際特許分類[7]: G06F 9/44

(21) 国際出願番号: PCT/JP2003/013302

(22) 国際出願日: 2003 年10 月17 日 (17.10.2003)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願 2002-305010
2002 年10 月18 日 (18.10.2002) JP

(71) 出願人 (米国を除く全ての指定国について): 株式会社アドバンテスト (ADVANTEST CORPORATION) [JP/JP]; 〒179-0071 東京都 練馬区 旭町 1 丁目 32 番 1 号 Tokyo (JP).

(71) 出願人 および
(72) 発明者: 前田 裕紀 (MAEDA,Hironori) [JP/JP]; 〒179-0071 東京都 練馬区 旭町 1 丁目 32 番 1 号 株式会社アドバンテスト内 Tokyo (JP).

(74) 代理人: 酒井 宏明 (SAKAI,Hiroaki); 〒100-0013 東京都 千代田区 霞が関三丁目 2 番 6 号 東京倶楽部ビルディング Tokyo (JP).

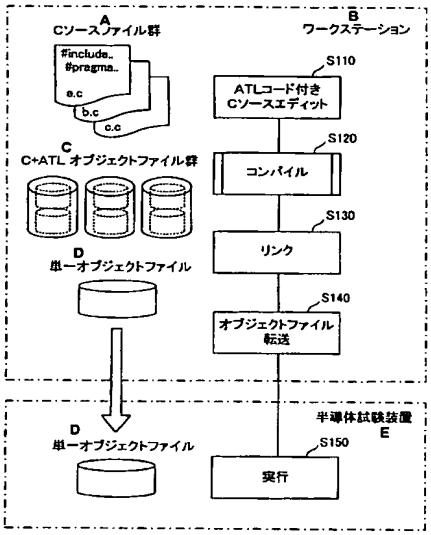
(81) 指定国 (国内): DE, KR, US.

添付公開書類:
— 国際調査報告書

[続葉有]

(54) Title: PROGRAM DEVELOPMENT SUPPORT DEVICE, PROGRAM EXECUTION DEVICE, COMPILE METHOD AND DEBUG METHOD

(54) 発明の名称: プログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法



A...C SOURCE FILES
B...WORKSTATION
S110...C SOURCE EDIT WITH ATL CODE
C... C+ATL OBJECT FILES
S120...COMPILE
S130...LINK
D...SINGLE OBJECT FILE
S140...OBJECT FILE TRANSFER
E...SEMICONDUCTOR TEST DEVICE
S150...EXECUTION

(57) Abstract: On a program development device (such as a workstation), created is a heterogeneous language source program where a description of a proprietary language source program (such as in the ATL) is embedded in a preprocessor describing section of a general purpose language source program (such as in the C language). From the heterogeneous language source program, the general purpose source program describing section and the particular specification language source program describing section are extracted, and respectively complied by their compilers to obtain their object codes. By combining the object codes a single object file is created.

(57) 要約: プログラム開発装置（例えば、ワークステーション）上で、汎用言語ソースプログラム（例えば、C言語）のプリプロセッサ記述部に、独自仕様言語ソースプログラム（例えば、ＡＴＬ）の記述を埋め込んだ異種言語混在ソースプログラムを作成する。そして、その異種言語混在ソースプログラムから汎用言語ソースプログラム記述部と独自仕様言語ソースプログラム記述部を取り出して、それぞれのコンパイラでコンパイルし、得られたそれぞれのオブジェクトコードを

[続葉有]

WO 2004/036420 A1

２文字コード及び他の略語については、定期発行される各*PCT*ガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

結合して、一つのオブジェクトファイルを生成する。

明 細 書

プログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法

技術分野

本発明は、半導体試験装置等の被制御装置に対する制御コマンドが記述された独自仕様言語プログラムと、独自仕様言語プログラムの実行ステップや独自仕様言語プログラムから得られたデータの処理ステップ等が記述された汎用言語プログラムとを開発するためのプログラム開発支援装置、それらプログラムを実行するプログラム実行装置、それらプログラムのコンパイル方法およびデバッグ方法に関し、特に、上記した独自仕様言語プログラムと汎用言語プログラムとが一つのファイル内に混在して記述された異種言語混在プログラムを開発するためのプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法に関する。

背景技術

膨大かつ多様な信号処理が求められる測定装置や通信装置などの電子機器の多くには、高性能なプロセッサが搭載されている。そのようなプロセッサ上で実行されるプログラム（ファームウェア）もまた、複雑でサイズが大きくなる傾向にある。必然と、これら電子機器に記録されているプログラムは、未知のバグを含んでいたり、機能の追加や改善を求められることが多い。

そのため、このような電子機器は、動作の設定／監視や上記したプログラムのアップデートを可能にするために、通信ケーブルを介して、外部のコンピュータと接続可能であることが多い。すなわち、外部のコンピュータから電子機器のプロセッサに対し、制御コマンドや制御プログラム自体を配信することが可能である。さらに換言すれば、外部のコンピュータ上で開発したプログラムのアルゴリ

ズムや設定内容に従って、電子機器を操作することが可能である。

そのような電子機器のうちでも特に半導体試験装置は、多種多様かつ独自仕様の半導体デバイスに対してそれぞれ個別のデバイステストプログラムを用意する必要がある特殊な測定装置である。特に、半導体試験装置のプロセッサ上で実行させるプログラムは、利用者（半導体デバイス製造メーカ）自らによって開発されるという形態が定着している。

ところが、半導体試験装置を筆頭とした特殊用途の電子機器は、搭載されるプロセッサも独自仕様のものが多く、必然とそのプロセッサが理解できるバイナリファイルも独自仕様に従っている。そのため、そのようなバイナリファイルの元となるソースファイルの作成に必要なプログラム言語の仕様や開発支援環境もまた特殊となり、プログラム開発者は、プログラム言語とともに開発支援環境の操作も習熟する必要があった。

このような背景から、近年においては、Ｃ言語やＪＡＶＡ（登録商標）等の汎用言語によって開発されたプログラムを実行することができる電子機器も登場してきている。ところが、そのような電子機器を採用することは、過去の独自仕様のプログラム資産を活用することができないという新たな問題を生み出す。

この問題を解決するために、データ処理やアルゴリズム全体が、Ｃ言語のような汎用言語プログラムで記述され、その汎用言語プログラム内から、電子機器に依存する独自仕様言語プログラムがサブルーチンとして呼び出されるという手法がある。以下に、この手法に従ったプログラムの開発と実行に関し、プログラムを実行する電子機器が半導体試験装置である場合を例に挙げて詳述する。

まず、半導体試験装置上でのプログラムの実行動作を理解するために、半導体試験装置とそのプログラム開発環境が説明される。半導体試験装置は、半導体メモリ、ロジックＩＣ、リニアＩＣなどの半導体デバイスに対して所定の動作試験を行う特殊な測定装置であり、一般にその装置構成は、上記した半導体デバイスの種別ごとに異なる。また、半導体試験装置への試験実行指示、試験結果取得およびプログラム開発は、通常、半導体試験装置に接続されたワークステーション

によって行なわれる。半導体試験は、例えば、特開２００１－１９５２７５号公報に開示されているように、半導体試験装置とワークステーションとで構成される半導体試験システムによって実現される。

第１０図は、従来の半導体試験システムの概略構成を示すブロック図であり、特に、上記したように異なる装置構成の半導体試験装置間において共通する構成を示すものである。第１０図において、半導体試験装置１００は、テスタプロセッサ１１０と、テスタ本体１２０と、テストヘッド１３０と、通信インターフェース１４０とを備えて構成される。

テスタプロセッサ１１０は、テスタ本体１２０との間で、制御コマンドの送信や試験データの送受信を行う手段であり、テスタ本体１２０の制御や後述するワークステーションとの間の通信を行うコントローラである。特に、テスタプロセッサ１１０は、内部に搭載したメモリ（図示略）に、ＯＳ（オペレーティングシステム）カーネル１１１を格納しており、デバイステストプログラムの起動や監視、メモリ管理を行うとともに、同じくメモリに格納された通信バスドライバ１１２やテスタバスドライバ１１３を介して、通信インターフェース１４０の監視／制御、テスタ本体１２０の制御、試験データの送受信を行う。

デバイステストプログラムは、上記した汎用言語プログラム１１４と独自仕様言語プログラム１１７とで構成されており、それら全体で、被測定デバイス１３１に対する機能試験やＤＣパラメトリック試験等の各種の試験を実施する手順を規定している。汎用言語プログラム１１４は、試験結果として取得した各種データを処理するコマンドを含んだステートメントと、デバイステストプログラム全体をどのように実行するかを示すコマンドを含んだステートメントとによって構成されており、ＯＳカーネル１１１上で直接実行可能なバイナリファイルである。

一方、独自仕様言語プログラム１１７は、テスタ本体１２０を制御するためのコマンドで構成されるオブジェクトファイルである。そのオブジェクトファイルは、過去の資産である独自仕様言語プログラムと同様に、独自仕様言語プログラム１１７に最適化されたカーネル上でのみ直接実行可能なバイナリファイルであ

る。そのため、独自仕様言語プログラム１１７をＯＳカーネル１１１上で実行させるにあたっては、実行用エミュレータ１１５による解釈処理が要求される。また、独自仕様言語プログラム１１７は、後述するワークステーション２００に対してのディスク・アクセス、キー入力、ディスプレイ表示といった入出力コマンドも含んでおり、それら入出力コマンドの実行にあたっては、実行用エミュレータ１１５による解釈に加え、さらにＩＯ制御用エミュレータ１１６による解釈処理が要求される。

テスタ本体１２０は、テスタプロセッサ１１０から送信される制御コマンドに従って、テストヘッド１３０に実装された被測定デバイス１３１に対し、機能試験やＤＣパラメトリック試験、ＲＦ試験（高周波試験）等の各種の試験を行う手段であり、レジスタ１２１、メモリ１２２、試験信号送受信部１２３を備えて構成される。レジスタ１２１は、テスタプロセッサ１１０内のテスタバスドライバ１１３との間で送受信される各種のデータを格納し、格納されたデータは、直接またはメモリ１２２を介して試験信号送受信部１２３に送信される。

また、試験信号送受信部１２３から出力されるデータは、一旦レジスタ１２１やメモリ１２２に格納された後、レジスタ１２１を介してテスタプロセッサ１１０内のテスタバスドライバ１１３に送信される。試験信号送受信部１２３は、パターン発生器やタイミング発生器、ＤＣユニット等の種々の試験ユニットから構成され、これら試験ユニットによって生成された試験信号を被測定デバイス１３１に入力するとともに、被測定デバイス１３１の出力ピンに現れるデータを取得する。

第１１図は、上記したワークステーション２００の概略構成を示すブロック図である。ワークステーション２００は、半導体試験装置１００内のテスタプロセッサ１１０に対してプログラムの転送や実行指示を行うコンソール端末の役割と、汎用言語プログラム１１４や独自仕様言語プログラム１１７の開発を支援するプログラム開発支援装置の役割とを担う。第１１図において、ワークステーション２００は、プロセッサ２２０と、通信インターフェース２４１と、ハードディス

ク装置２４２と、マウス２４３と、キーボード２４４と、ディスプレイ装置２４５とを備えて構成される。

プロセッサ２２０は、内部に搭載したメモリ（図示略）に、ＯＳ（オペレーティングシステム）カーネル２２１を格納しており、種々のプログラムの起動や監視、メモリ管理を行うとともに、同じくメモリに格納される通信バスドライバ２２３、ハードディスクドライバ２２４、マウスドライバ２２５、キーボードドライバ２２６、ディスプレイドライバ２２７を介して、通信インターフェース２４１の監視や制御、ハードディスク装置２４２との間でのプログラムやデータの読み込み／書き込み、マウス２４３やキーボード２４４からの入力情報の取得、ディスプレイ装置２４５への表示情報の出力を行う。ここで特に、通信インターフェース２４１は、通信ケーブル（図示せず）を介して、第１０図に示した通信インターフェース１４０と接続されており、ワークステーション２００と半導体試験装置１００との間の通信を可能にさせる。

また、ＯＳカーネル２２１は、ＧＵＩ（Ｇｒａｐｈｉｃａｌ　Ｕｓｅｒ　Ｉｎｔｅｒｆａｃｅ）処理部２２２を有する。図示されるエディタ２２８、汎用言語コンパイラ２２９、リンカ２３３、汎用言語デバッガ２３１、独自仕様言語コンパイラ２３０、独自仕様言語デバッガ２３２などの各種プログラムは、ディスプレイ装置２４５上に表示されるウィンドウ画面単位で実行されることが可能である。このワークステーション２００は、汎用的なコンピュータの装置構成と同等である。よって、上記した各種ドライバや各種プログラムは、通常はハードディスク装置２４２に記憶されており、ＯＳカーネル２２１に従って必要に応じて上記メモリに読み込まれて実行される。

次に、半導体試験装置１００とワークステーション２００とで構成される半導体試験システムにおいて、デバイステストプログラムの開発と実行の流れが説明される。第１２図は、従来のデバイステストプログラムの開発と実行手順を示したフローチャートである。ここでは、デバイステストプログラムが、上記したように汎用言語プログラムと独自仕様言語プログラムとで構成されており、汎用言

語プログラムとしてC言語を採用し、独自仕様言語プログラムとしてＡＴＬ（アドバンテスト社独自規格）を採用した場合を例に挙げる。

まず、プログラム開発者は、ワークステーション２００上においてエディタ２２８を起動させ、C言語によるソースプログラムを作成する（ステップＳ３０１）。このソースプログラムは、上述したように、デバイステストプログラム全体のアルゴリズムを記述しており、シーケンス処理の所望の位置で、ＡＴＬで記述されたオブジェクトプログラムを呼び出して実行したり、その実行によって得られた試験結果データを処理する手順を規定する。

C言語によるソースプログラムの作成を終えると、プログラム開発者は、Cコンパイラ（汎用言語コンパイラ２２９に相当）に対し、作成したソースプログラムのファイル（以下、必要なヘッダファイル等を含めてCソースファイルと称する。）を指定してコンパイルを実行させる（ステップＳ３０２）。コンパイル処理では、まず構文チェックが行なわれ、構文エラーが生じた場合には、プログラム開発者は、エディタ２２８でエラー箇所を修正し、再度コンパイルの実行を指示する。エラーがない場合には、上記したCソースファイルを機械語に翻訳したオブジェクトファイル（以下、Cオブジェクトファイルと称する。）が生成される。

ステップＳ３０１で作成された複数のCソースファイルに対し、ステップＳ３０２が完了すると、プログラム開発者は、リンカ２３３に対し、生成された複数のCオブジェクトファイルやその他に必要なライブラリファイルを指定してリンクを実行させる（ステップＳ３０３）。このリンクによって、半導体試験装置１００のテスタプロセッサ１１０上で直接実行可能な単一のCオブジェクトファイルが生成される。

また、プログラム開発者は、C言語によるオブジェクトファイルの生成と並行して、ワークステーション２００上においてエディタ２２８を起動させ、ＡＴＬによるソースプログラムを作成する（ステップＳ４０１）。このソースプログラムは、上述したように、半導体試験装置１００を制御するための制御コマンドを

記述する。

ＡＴＬによるソースプログラムの作成が終えると、プログラム開発者は、ＡＴＬコンパイラ（独自仕様言語コンパイラ２３０に相当）に対し、作成したソースプログラムのファイル（以下、ＡＴＬソースファイルと称する。）を指定してコンパイルを実行させる（ステップＳ４０２）。なお、このコンパイル処理でも、上記したステップＳ３０２と同様に、まず構文チェックが行なわれ、構文エラーが生じた場合には、プログラム開発者は、エディタ２２８でエラー箇所を修正し、再度コンパイルの実行を指示する。エラーがない場合には、上記したＡＴＬソースプログラムは、上記したＣオブジェクトファイルで表わされる機械語（ある特定のテスタプロセッサで理解できる機械語）とは異なる旧テスタプロセッサ仕様の機械語に翻訳され、オブジェクトファイル（以下、ＡＴＬオブジェクトファイルと称する。）が生成される。

このような手順によって、単一Ｃオブジェクトファイルとａｔｌオブジェクトファイル群が用意されると、プログラム開発者は、ワークステーション２００上において、半導体試験装置１００との通信を可能にするコントロールプログラムを起動し、そのコントロールプログラムを用いて単一Ｃオブジェクトファイルとａｔｌオブジェクトファイル群を半導体試験装置１００のテスタプロセッサ１１０へと転送する（ステップＳ３０４，ステップＳ４０３）。

続いて、プログラム開発者は、上記したコントロールプログラムに対して、単一Ｃオブジェクトファイルの実行指示を与える（ステップＳ３０５）。これにより、半導体試験装置１００のテスタプロセッサ１１０は、単一Ｃオブジェクトファイルに記述されたアルゴリズムに従って、ＡＴＬオブジェクトファイルの実行→テスタ本体１２０の所望の試験ユニットの稼動→被測定デバイス１３１から得られた試験結果の取得→データ処理を繰り返す。この際、データ処理によって適宜加工等が行われた試験結果は、半導体試験装置１００の通信インターフェース１４０、通信ケーブル、ワークステーション２００の通信インターフェース２４１を介して、上記したコントロールプログラムで受け取ることができ、そのコン

トロールプログラムに割り当てられたウィンドウ画面上に表示される。

ここで、プログラム開発者は、試験結果が明らかに異常である場合等の不具合を発見した場合、デバイステストプログラムに論理的なエラーが含まれていると判断し、ワークステーション２００上において汎用言語デバッガ２３１を起動させ、Ｃソースファイル中の所定のステートメントにブレークポイントを設定する。そして、プログラム開発者がデバッグ開始を指示することにより、汎用言語デバッガ２３１は、再度、上記したステップＳ３０２～Ｓ３０５の手順によって単一Ｃオブジェクトファイルを実行し、実行されたステートメントが、設定したブレークポイントに達したことを検出すると、ブレークしたステートメントの段階で有効となっている変数を表示する。プログラム開発者は、この変数の確認によって論理的なエラーを発見すると、エディタ２２８を起動させ、Ｃソースファイルを適宜修正して上記したステップＳ３０２～Ｓ３０５の手順を繰り返す。

一方、プログラム開発者は、汎用言語デバッガ２３１によってＣソースファイルの論理的なエラーを発見できないときは、続いて、独自仕様言語デバッガ２３２を起動させ、ＡＴＬソースファイル中の所定のステートメントにブレークポイントを設定し、上記同様のデバッグ処理を行う。

しかしながら、上記したように、被制御装置（上記例では半導体試験装置）上で実行させるプログラムが汎用言語と独自仕様言語のように異種言語で記述される場合には、独自仕様言語による過去のプログラム資産を活用することができるものの、双方のプログラムの開発段階において、それぞれ個別のソースファイルが必要とされる。上記した例では、ＣソースファイルとＡＴＬソースファイルとをそれぞれ別ファイルとして用意する必要がある。簡単に言えば、汎用言語で記述されたソースファイル中に独自仕様言語に作成されたオブジェクトファイルの呼び出しを埋め込んだ場合には、少なくとも２つのソースファイルが必要となる。特に、ある一つの汎用言語ソースファイルに対しては、ある特定の独自仕様言語ソースファイルが対応するため、これらファイルはひとまとめで管理しなければならない。

さらに、双方のソースファイルの内容から、関連する異種言語のソースファイルを特定することは困難であり、一度管理を誤ると、対応関係を再度見出すのに多くの時間と手間を要するという問題があった。このことから、エディタ上でのソースファイルの修正作業や他のソースファイルの部分的な流用については慎重にならざるを得ず、上記問題は、プログラム開発者にとってもミスを増やす原因となっていた。

また、一つの実行プログラムに対して、汎用言語ソースファイルと独自仕様言語ソースファイルとを用意することから、必然と、それらをコンパイルすることによって同じ数だけオブジェクトファイルが生成される。これは、上記した管理がさらに複雑になることを意味する。このように、従来では、一つの実行プログラムに対して異種言語の複数のソースファイルおよびオブジェクトファイルが存在したため、ファイル管理が煩雑となり、プログラムの開発効率を低下させるという問題があった。

本発明は上記に鑑みてなされたものであって、汎用言語ソースファイルのプリプロセッサ記述部に独自仕様言語のソースファイルを埋め込むことによって、独自仕様言語のプログラムによる過去の資産を活用できるとともに、一つの実行ファイルに対して必要なソースファイルおよびオブジェクトファイルの数を大幅に減少させることのできるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法を提供することを目的とする。

発明の開示

上記目的を達成するために、本発明にかかるプログラム開発支援装置は、汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置上で実行可能なプログラムファイルを作成するためのプログラム開発支援装置において、前記独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイル手段（後述する独自仕様言語コンパイラ

３０に相当）と、前記異種言語混在ソースプログラム内の前記汎用言語ソースプログラム記述部分をコンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイル手段（後述する汎用言語コンパイラ２９に相当）と、前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出し、抽出した独自仕様言語ソースプログラムを指定して前記独自仕様言語コンパイル手段を実行させるとともに、前記異種言語混在ソースプログラムを指定して前記汎用言語コンパイル手段を実行させ、得られた独自仕様言語オブジェクトコードと汎用言語オブジェクトコードを結合してオブジェクトファイルを生成する統合コンパイル手段（後述する統合コンパイラ３４に相当）と、前記統合コンパイル手段によって生成された少なくとも一つのオブジェクトファイルから前記プログラムファイルを生成するリンク手段（後述するリンカ３３に相当）と、を備えたことを特徴としている。

また、本発明にかかるプログラム実行装置は、汎用言語ソースプログラムのオブジェクトコードと独自仕様言語ソースプログラムのオブジェクトコードが混在したプログラムファイルを実行するプログラム実行装置（後述する半導体試験装置１１に相当）において、前記プログラムファイルの実行開始時に、前記汎用言語ソースプログラムのオブジェクトコードと前記独自仕様言語ソースプログラムのオブジェクトコードをメモリにロードすることを特徴としている。

また、本発明にかかるコンパイル方法は、汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置上で実行可能なプログラムファイルを作成するためのコンパイル方法において、前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出する独自仕様言語ソースプログラム抽出ステップ（後述するステップＳ１２１に相当）と、抽出された独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイルステップ（後述するステップＳ１２３に相当）と、前記異種言語混在ソースプログラムのうちの前記汎用言語ソースプログラム記述部分を

コンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイルステップ（後述するステップＳ１２２に相当）と、前記独自仕様言語オブジェクトコードと前記汎用言語オブジェクトコードを結合してオブジェクトファイルを生成するオブジェクトファイル生成ステップ(後述するステップＳ１２４に相当)と、前記オブジェクトファイル生成ステップによって生成された少なくとも一つのオブジェクトファイルから前記プログラムファイルを生成するリンクステップ（後述するステップＳ１３０に相当）と、を含んだことを特徴としている。

また、本発明にかかるデバッグ方法は、汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから作成された所定のプログラム実行装置上で実行可能なプログラムファイルをデバッグするためのデバッグ方法において、前記異種言語混在ソースプログラム内のステートメントにブレークポイントを設定するブレークポイント設定ステップと、前記プログラムファイルの実行時に前記ブレークポイントで該プログラムファイルを停止させるとともに、停止したプログラムファイルの前記ステートメントが前記汎用言語ソースプログラムに属する場合に汎用言語デバッガを起動し（後述するステップＳ２０３に相当）、停止したプログラムファイルの前記ステートメントが前記独自仕様言語ソースプログラムに属する場合に独自仕様言語デバッガを起動する（後述するステップＳ２０６に相当）デバッガ起動ステップと、前記汎用言語デバッガと前記独自仕様言語デバッガとから得られたデバッグ情報（後述するステップＳ２０４，ステップＳ２０７に相当）を共通するウィンドウ画面に表示するデバッグ情報表示ステップ（後述するステップＳ２０５に相当）と、を含んだことを特徴としている。

また、本発明にかかるコンピュータ読み取り可能な記録媒体は、上記コンパイル方法をコンピュータに実行させることを特徴としている。

また、本発明にかかるコンピュータ読み取り可能な記録媒体は、上記デバッグ方法をコンピュータに実行させることを特徴としている。

図面の簡単な説明

第１図は、実施の形態にかかる半導体試験システムの概略構成を示すブロック図であり、第２図は、デバイステストプログラムの開発と実行手順を示したフローチャートであり、第３図は、Ｃ＋ＡＴＬソースプログラムの記述例を示す図であり、第４図は、統合コンパイラによるコンパイル処理を説明するためのフローチャートであり、第５図は、ＡＴＬソースファイルの生成処理を説明するための説明図であり、第６図は、Ｃ＋ＡＴＬオブジェクトファイルの構成を示す図であり、第７図は、デバッガ選択ルーチンを示すフローチャートであり、第８図は、ＡＴＬ記述部内でブレークした状態の統合デバッガの実行画面の例を示す図であり、第９図は、Ｃ言語記述部内でブレークした状態の統合デバッガの実行画面の例を示す図であり、第１０図は、従来の半導体試験システムの概略構成を示すブロック図であり、第１１図は、従来の半導体試験システムのワークステーションの概略構成を示すブロック図であり、第１２図は、従来のデバイステストプログラムの開発と実行手順を示したフローチャートである。

発明を実施するための最良の形態

以下に、本発明にかかるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法の実施の形態を図面に基づいて詳細に説明する。この実施の形態により本発明は限定されない。

この実施の形態では、本発明の特徴の理解を容易にするために、上述した従来技術の説明と同様に、本発明を半導体試験装置とワークステーションとから構成される半導体試験システムに適用した場合の形態が説明される。具体的には、本発明にかかるプログラム開発支援装置が、半導体試験システムのワークステーションに相当し、本発明にかかるプログラム実行装置が半導体試験システムの半導体試験装置に相当し、本発明にかかるコンパイル方法およびデバッグ方法が上記半導体試験システム上でのコンパイル方法およびデバッグ方法に相当する。

第１図は、本実施の形態にかかる半導体試験システムを示すブロック図である。

第１図に示す半導体試験システムは、通信ケーブルで接続されたワークステーション１０と半導体試験装置１１とを備えて構成される。

ワークステーション１０の基本構成は、第１１図に示された従来の半導体試験システムのワークステーション２００と同じであり、第１図中のプロセッサ２０と、通信インターフェース４１と、ハードディスク装置４２と、マウス４３と、キーボード４４と、ディスプレイ装置４５は、それぞれ第１１図に示したプロセッサ２２０、通信インターフェース２４１、ハードディスク装置２４２、マウス２４３、キーボード２４４、ディスプレイ装置２４５にそれぞれ対応する。

また、プロセッサ２０内部において、メモリ（図示略）に格納されるＯＳカーネル２１、ＧＵＩ処理部２２、通信バスドライバ２３、ハードディスクドライバ２４、マウスドライバ２５、キーボードドライバ２６、ディスプレイドライバ２７、エディタ２８、汎用言語コンパイラ２９、独自仕様言語コンパイラ３０、汎用言語デバッガ３１、独自仕様言語デバッガ３２、リンカ３３もまた、それぞれ第１１図に示したＯＳカーネル２２１、ＧＵＩ処理部２２２、通信バスドライバ２２３、ハードディスクドライバ２２４、マウスドライバ２２５、キーボードドライバ２２６、ディスプレイドライバ２２７、エディタ２２８、汎用言語コンパイラ２２９、独自仕様言語コンパイラ２３０、汎用言語デバッガ２３１、独自仕様言語デバッガ２３２、リンカ２３３にそれぞれ対応する。

本実施の形態で説明するワークステーション１０は、汎用言語コンパイラ２９と独自仕様言語コンパイラ３０の上位アプリケーションに位置する統合コンパイラ３４を有している点で、従来のワークステーション２００と異なる。換言すれば、このワークステーション１０は、統合コンパイラ３４から、汎用言語コンパイラ２９と独自仕様言語コンパイラ３０をそれぞれ利用することができる。

さらに、本実施の形態で説明するワークステーション１０は、汎用言語デバッガ３１と独自仕様言語デバッガ３２の上位アプリケーションに位置する統合デバッガ３５を有している点で、従来のワークステーション２００と異なる。すなわち、統合コンパイラ３４と同様に、このワークステーション１０は、統合デバッ

ガ３５から、汎用言語デバッガ３１と独自仕様言語デバッガ３２をそれぞれ利用することができる。

また、第１図に示す半導体試験システムでは、半導体試験装置１１はワークステーション１０に接続されているが、この半導体試験装置１１の内部構成は第１０図に示した従来の半導体試験装置１００と同様である。しかし、この半導体試験装置１１は、ワークステーション１０において開発されるプログラムの態様に依存して、テスタプロセッサの動作が従来と一部異なる。

次に、この半導体試験システムにおいて、デバイステストプログラムの開発と実行の流れが説明される。第２図は、本実施の形態におけるデバイステストプログラムの開発と実行手順を示したフローチャートである。ここでは、第１２図の説明と同様に、デバイステストプログラムが、汎用言語プログラムと独自仕様言語プログラムとで構成されており、汎用言語プログラムとしてＣ言語を採用し、独自仕様言語プログラムとしてＡＴＬ（アドバンテスト社独自規格）を採用した場合を例に挙げる。

まず、プログラム開発者は、ワークステーション１０上においてエディタ２８を起動させ、ソースプログラムを作成する（ステップＳ１１０）。このソースプログラムは、汎用言語であるＣ言語で記述されるが、第１２図で説明したＣソースファイルの内容とは異なり、プリプロセッサ記述部に、独自仕様言語であるＡＴＬソースファイルの内容を記述する。このソースプログラムをＣ＋ＡＴＬソースプログラムと称する。

第３図は、Ｃ＋ＡＴＬソースプログラムの記述例を示す図である。なお、第３図において、左端に配置された「数字：」は、説明の便宜のために用いられる行番号を示すが、実際のプログラム動作においては無視される。以下のＣ＋ＡＴＬソースプログラムの内容の説明では、この行番号によって各ステートメントが参照される。

Ｃ言語コンパイラは、先頭が＃で続くステートメントをプリプロセッサコマンドであると認識する。第３図に示すＣ＋ＡＴＬソースプログラムでは、行番号１

の＃ｉｎｃｌｕｄｅ、行番号３，４，５の＃ｐｒａｇｍａがそのプリプロセッサコマンドに相当する。＃ｉｎｃｌｕｄｅは、ヘッダファイル「ＡＴ／ｈｙｂｒｉｄ．ｈ」の記述内容を単にその位置に展開するコマンドであり、行番号１０以降に続くメイン関数内において必要となる内容である。

一方、＃ｐｒａｇｍａは、Ｃ言語との全般的な互換性を保ちながら、マシンまたはＯＳに固有の機能を実現する特別なプリプロセッサコマンドである。よって、＃ｐｒａｇｍａは定義上、マシンまたはＯＳに固有であり、通常、コンパイラごとに異なる。＃ｐｒａｇｍａは、本来、プリプロセッサに新しい機能を与えたり、インプリメントに依存する情報をコンパイラに与えるために利用されるが、本実施の形態で作成されるＣ＋ＡＴＬソースプログラムでは、＃ｐｒａｇｍａによって渡されるトークンとして、独自仕様言語であるＡＴＬの記述を埋め込む。この＃ｐｒａｇｍａによって渡されるＡＴＬの記述の処理は後述される。

第３図に示すＣ＋ＡＴＬソースプログラムにおいて、行番号１０～２８に記述されている内容は、従来の半導体試験システムのワークステーション上において作成される内容と同じであり、その記述の中において、ＡＴＬオブジェクトファイルの呼び出しやデータ処理が規定される。

Ｃ＋ＡＴＬソースプログラムの作成が終えると、プログラム開発者は、統合コンパイラ３４に対し、作成したＣ＋ＡＴＬソースプログラムのファイル（以下、必要なヘッダファイル等を含めてＣ＋ＡＴＬソースファイルと称する。）を指定してコンパイルを実行させる（ステップＳ１２０）。このコンパイルの実行に際しては、まず構文チェックが行なわれ、構文エラーが生じた場合には、プログラム開発者は、エディタ２８でエラー箇所を修正し、再度コンパイルの実行を指示する。エラーがない場合に初めて、オブジェクトファイルを生成するためのコンパイル処理が開始される。

第４図は、統合コンパイラ３４によるコンパイル処理を説明するためのフローチャートである。統合コンパイラ３４は、ステップＳ１１０において作成されたＣ＋ＡＴＬソースファイルにおいて構文エラーを発見できなかったときは、続い

て、そのＣ＋ＡＴＬソースファイルからＡＴＬ記述部を抽出し、ＡＴＬソースファイルを生成する（ステップＳ１２１）。第５図は、このＡＴＬソースファイルの生成処理を説明するための説明図である。ＡＴＬソースファイルの生成処理は、上述したように、Ｃ＋ＡＴＬソースファイルのプリプロセッサ記述部から、＃ｐｒａｇｍａを識別し、＃ｐｒａｇｍａの後に続くトークンを解析することから始まる。第５図に示す例では、＃ｐｒａｇｍａの直後のａｔｌが、その後に続く情報がＡＴＬ記述部であることを示すキーワードとなる。

第５図の例を用いて具体的な説明を行うと、統合コンパイラ３４は、行番号３において、＃ｐｒａｇｍａ　ａｔｌを認識すると、その後に続くキーワードのｎａｍｅを取り出し、そのキーワードｎａｍｅに続くダブルクォーテーションで囲まれた記述、すなわちＳＡＭＰＬＥがプログラム名であると解釈する。この解釈によって、ＡＴＬソースファイルの先頭部に、ＰＲＯ　ＳＡＭＰＬＥが挿入される。つぎに、統合コンパイラ３４は、行番号４において、＃ｐｒａｇｍａ　ａｔｌを認識すると、その後に続くキーワードのｓｏｃｋｅｔを取り出し、そのキーワードｓｏｃｋｅｔに続くダブルクォーテーションで囲まれた記述、すなわちＳＳＯＣが使用するＡＴＬのソケットプログラム名であると解釈する。この解釈によって、ＡＴＬソースファイル内のＰＲＯ　ＳＡＭＰＬＥの後にＳＳＯＣが挿入される。

さらに、統合コンパイラ３４は、行番号５において、＃ｐｒａｇｍａ　ａｔｌを認識すると、その後に続くキーワードのｐｒｏｃを取り出し、そのキーワードｐｒｏｃの直後の文字列とその後に続くダブルクォーテーションで囲まれた記述、すなわち、

Ｐ１（ＡＲＧ１，ＡＲＧ２（２））　”｛ＷＲＩＴＥ　”ＡＲＧ１＝”，ＡＲＧ１，／ＷＲＩＴＥ　”ＡＲＧ２＝”，ＡＲＧ２，／｝”

が関数定義であると解釈する。この解釈によって、ＡＴＬソースファイルに、

Ｐ１：ＡＲＧＵＭＥＮＴ（ＡＲＧ１，ＡＲＧ２（２））

ＷＲＩＴＥ　”ＡＲＧ１＝”，ＡＲＧ１；／

WRITE "ARG2=", ARG2, /

GOTO CONTINUE

が追記される。

そして、統合コンパイラ３４は、Ｃ＋ＡＴＬソースファイルにおいて、＃ｐｒａｇｍａ　ａｔｌを発見することができないまま、Ｃ＋ＡＴＬソースファイルの最終行（行番号２８）に達すると、ＡＴＬソースファイルの最後にＥＮＤを挿入してそのＡＴＬソースファイルの作成を終える。

ＡＴＬソースファイルの作成が終わると、統合コンパイラ３４は、Ｃ＋ＡＴＬソースファイルのＣ言語記述部のコンパイル、すなわちＣオブジェクトコードの生成を行う（ステップＳ１２２）。このコンパイルは、通常のＣコンパイラ（汎用言語コンパイラ２９に相当する。）によって処理され、上記した＃ｐｒａｇｍａの記述部分は無視される。統合コンパイラ３４がＣコンパイラを呼び出して処理させるとしているが、統合コンパイラ３４自体にＣコンパイラの機能を取り入れ、上記したＡＴＬソースファイルの生成と並行して、Ｃオブジェクトコードの生成が行われてもよい。この場合、第１図に示した汎用言語コンパイラ２９は不要となる。

Ｃオブジェクトコードの生成が終わると、統合コンパイラ３４は、ステップＳ１２１において生成されたＡＴＬソースファイルのコンパイル、すなわちＡＴＬオブジェクトコードの生成を行う（ステップＳ１２３）。このコンパイルは、ＡＴＬコンパイラ（独自仕様言語コンパイラ３０に相当する。）の呼び出しによって実行され、第１２図のステップＳ４０２と同様に、上記したＣオブジェクトコードで表わされる機械語（ある特定のテスタプロセッサで理解できる機械語）とは異なる旧テスタプロセッサ仕様の機械語への翻訳を行う。

ＡＴＬオブジェクトコードの生成が終わると、統合コンパイラ３４は、ステップＳ１２２において生成されたＣオブジェクトコードに、ステップＳ１２１において生成されたＡＴＬオブジェクトコードを結合し、さらにそのＡＴＬオブジェクトコードが格納された位置情報（ＡＴＬオブジェクトコード開始位置）を付加

したオブジェクトファイル（以下、Ｃ＋ＡＴＬオブジェクトファイルと称する。）を生成する（ステップＳ１２４）。第６図は、このＣ＋ＡＴＬオブジェクトファイルの構成を示す図である。第６図に示すように、Ｃ＋ＡＴＬオブジェクトファイルは、Ｃオブジェクトコードに続いてＡＴＬオブジェクトコードが配置される。同図において、ＡＴＬオブジェクトコードの位置情報等の付加情報の図示は省略されている。

この統合コンパイラ３４によるコンパイル処理は、上記同様に作成された複数のＣ＋ＡＴＬソースファイルに対して行われ、これにより複数のＣ＋ＡＴＬオブジェクトファイルが用意される。このようにして統合コンパイラ３４によるコンパイル処理が終わると、プログラム開発者は、リンカ３３に対し、生成された複数のＣ＋ＡＴＬオブジェクトファイルやその他に必要なライブラリファイルを指定してリンクを実行させる（第２図のステップＳ１３０）。

リンカ３３は、複数のＣ＋ＡＴＬオブジェクトファイルやその他に必要なライブラリファイルに加え、上記各Ｃ＋ＡＴＬオブジェクトファイルからＡＴＬオブジェクトコード部分をロードするためのロードプログラムを用意し、それらをリンクすることによって、半導体試験装置１１のテスタプロセッサ上で直接実行可能な単一オブジェクトファイルを生成する。

このような手順によって、単一オブジェクトファイルが用意されると、プログラム開発者は、ワークステーション１０上において、半導体試験装置１１との通信を可能にするコントロールプログラムを起動し、そのコントロールプログラムを用いて上記単一オブジェクトファイルを半導体試験装置１１のテスタプロセッサへと転送する（ステップＳ１４０）。

続いて、プログラム開発者は、上記したコントロールプログラムに対して、単一オブジェクトファイルの実行指示を与える（ステップＳ１５０）。これにより、半導体試験装置１１のテスタプロセッサは、まず、単一オブジェクトファイルに含まれているロードプログラムに従って、同単一オブジェクトファイル内に配置されているＣオブジェクトコードとＡＴＬオブジェクトコードをメモリ上にロー

ドする。続いて、テスタプロセッサは、ロードされたＣオブジェクトコードに記述されたアルゴリズムに従って、同じくロードされているＡＴＬオブジェクトコードの実行→テスタ本体の所望の試験ユニットの稼動→被測定デバイスから得られた試験結果の取得→データ処理を繰り返す。この際、データ処理によって適宜加工等が行われた試験結果は、従来と同様に、半導体試験装置１１の通信インターフェース、通信ケーブル、ワークステーション１０の通信インターフェース４１を介して、上記したコントロールプログラムで受け取ることができ、そのコントロールプログラムに割り当てられたウィンドウ画面上に表示される。

なお、ここでは、単一オブジェクトファイル内に、ロードプログラムが含まれているとしたが、半導体試験装置１１のテスタプロセッサ内において、あらかじめそのロードプログラムが読み込まれ、ワークステーション１０からの実行指示に従って、最初にこのロードプログラムが起動されてもよい。

次に、本実施の形態における半導体試験システムのデバック処理が説明される。プログラム開発者は、ステップＳ１５０の実行によって得た試験結果が、明らかに異常である場合等の不具合を発見した場合、従来と同様、デバイステストプログラムに対してデバック処理を行う。まず、プログラム開発者は、ワークステーション１０上において、統合デバッガ３５を起動させ、Ｃ＋ＡＴＬソースファイル中の所定のステートメントにブレークポイントを設定する。

そして、プログラム開発者がデバッグ開始を指示することによって、統合デバッガ３５は、再度、上記したステップＳ１２０～Ｓ１５０の手順によって単一オブジェクトファイルを実行し、実行されたステートメントが、設定したブレークポイントに達したことを検出すると、Ｃデバッガ（汎用言語デバッガ３１に相当する。）とＡＴＬデバッガ（独自仕様言語デバッガ３２に相当する。）のいずれかのデバッガを起動させるかのデバッガ選択ルーチンを実行する。

第７図は、このデバッガ選択ルーチンを示すフローチャートである。統合デバッガ３５は、単一オブジェクトファイル中において順次実行されているステートメントがブレークポイントに到達すると、そのブレークポイントが設定されたス

テートメントを表示する（ステップＳ２０１）。そして、そのステートメントがＡＴＬオブジェクトコード部に相当するならば（ステップＳ２０２：Ｙｅｓ）、ＡＴＬデバッガを起動し（ステップＳ２０６）、ＡＴＬデバッガからブレークしたステートメントに含まれる変数等のデバッグ情報を取得する（ステップＳ２０７）。なお、ＡＴＬデバッガは、上記したブレークポイント設定時に、統合デバッガ３５によってＡＴＬオブジェクトコード部のブレークポイント設定情報を取得している。

統合デバッガ３５は、ＡＴＬデバッガからデバッグ情報を取得すると、ブレーク時に有効となっている指定変数（シンボル）を表示する（ステップＳ２０５）。第８図は、統合デバッガ３５の実行画面の例を示す図であり、特にＡＴＬ記述部内でブレークした状態を示している。第８図において、統合デバッガ３５は、実行ウィンドウ５０内に、ウィンドウ表示に標準として付加されるタイトルバーやメニューバーに加え、ブレークポイント設定領域５１、ソース表示領域５２、シンボル表示領域５３を有する。第８図では、Ｃ＋ＡＴＬソースプログラム中のＡＴＬ記述部内でブレークした状態として、ブレークポイントが設定された行番号５のステートメントがソース表示領域５２内に表示されるとともに、シンボル表示領域５３内に、そのステートメントで使用されている変数と変数に格納された値が表示されている。

一方、統合デバッガ３５は、ブレークしたステートメントがＣオブジェクトコード部に相当するならば（ステップＳ２０２：Ｎｏ）、Ｃデバッガを起動し（ステップＳ２０３）、Ｃデバッガからブレークしたステートメントに含まれる変数等のデバッグ情報を取得する（ステップＳ２０４）。なお、Ｃデバッガは、上記したブレークポイント設定時に、統合デバッガ３５によってＣオブジェクトコード部のブレークポイント設定情報を取得している。

統合デバッガ３５は、Ｃデバッガからデバッグ情報を取得すると、ブレーク時に有効となっている指定変数（シンボル）を表示する（ステップＳ２０５）。第９図は、統合デバッガ３５の実行画面の例を示す図であり、特にＣ言語記述部内

でブレークした状態を示している。第９図に示す統合デバッガ３５の実行ウィンドウ５０は、第８図と同様な構成である。第９図では、Ｃ＋ＡＴＬソースプログラム中のＣ言語記述部内でブレークした状態として、ブレークポイントが設定された行番号１５のステートメントがソース表示領域５２内に表示されるとともに、シンボル表示領域５３内に、そのステートメントで使用されている変数と変数に格納された値が表示されている。

プログラム開発者は、この統合デバッガ３５のウィンドウ画面上に表示された変数の値を確認することによって論理的なエラーを発見した場合、エディタ２８を起動させ、Ｃ＋ＡＴＬソースファイルを適宜修正して上記したステップＳ１２０～Ｓ１５０の手順を繰り返す。

以上に説明したとおり、実施の形態にかかる半導体試験システム、すなわち本発明にかかるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法によれば、独自仕様言語プログラムであるＡＴＬソース自体を汎用言語プログラムであるＣ言語ソース内に埋め込むことによって、従来別管理されていた双方のソースを一つのＣ＋ＡＴＬソースファイルとして取り扱うことかできるので、ファイル管理を楽にし、プログラムの開発効率を向上させることができる。

なお、本実施の形態では、本発明にかかるプログラム開発支援装置とプログラム実行装置をそれぞれワークステーションと半導体試験装置に適用した形態を例示したが、プログラム開発支援装置を汎用的なコンピュータシステムとし、プログラム実行装置をそのコンピュータシステムと通信可能な測定装置や制御装置に適用することができるのは言うまでもない。

以上に説明したように、本発明にかかるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法によれば、独自仕様言語プログラム自体を汎用言語プログラム内に埋め込むことで、従来別管理されていた双方のソースプログラムを、ソースファイルのみでなく、コンパイルによって作成されるオブジェクトファイルの段階においても一つのファイルとして取り扱うことが

できるので、ファイル管理を楽にし、プログラムの開発効率を向上させることができるという効果を奏する。

産業上の利用可能性

以上のように、本発明にかかるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法は、高性能な電子機器のプログラム（ファームウェア）を効率的に開発し、かつそのプログラムの管理を容易にするのに有用であり、特に、半導体試験装置のプログラムの開発および管理に適している。

請 求 の 範 囲

1． 汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置上で実行可能なプログラムファイルを作成するためのプログラム開発支援装置において、

前記独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイル手段と、

前記異種言語混在ソースプログラム内の前記汎用言語ソースプログラム記述部分をコンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイル手段と、

前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出し、抽出した独自仕様言語ソースプログラムを指定して前記独自仕様言語コンパイル手段を実行させるとともに、前記異種言語混在ソースプログラムを指定して前記汎用言語コンパイル手段を実行させ、得られた独自仕様言語オブジェクトコードと汎用言語オブジェクトコードを結合してオブジェクトファイルを生成する統合コンパイル手段と、

前記統合コンパイル手段によって生成された少なくとも一つのオブジェクトファイルから前記プログラムファイルを生成するリンク手段と、

を備えたことを特徴とするプログラム開発支援装置。

2． 前記統合コンパイル手段は、前記オブジェクトファイルに、前記独自仕様言語オブジェクトコードおよび／または前記汎用言語オブジェクトコードのコード位置情報を付加することを特徴とする請求の範囲第１項に記載のプログラム開発支援装置。

3． 前記プログラムファイルを前記プログラム実行装置に転送するプログラム

転送手段を備えたことを特徴とする請求の範囲第１項に記載のプログラム開発支援装置。

４．　前記プログラム実行装置に対して該プログラム実行装置に転送されたプログラムファイルを実行させる指示を与えるプログラム実行装置コントロール手段を備えたことを特徴とする請求の範囲第３項に記載のプログラム開発支援装置。

５．　前記異種言語混在ソースプログラム内のステートメントにブレークポイントを設定するブレークポイント設定手段と、

前記プログラムファイルの実行時に前記ブレークポイントで該プログラムファイルを停止させるとともに、停止したプログラムファイルの前記ステートメントが前記汎用言語ソースプログラムに属する場合に汎用言語デバッガを起動し、停止したプログラムファイルの前記ステートメントが前記独自仕様言語ソースプログラムに属する場合に独自仕様言語デバッガを起動することを特徴とする請求の範囲第１項に記載のプログラム開発支援装置。

６．　前記汎用言語デバッガと前記独自仕様言語デバッガとから得られたデバッグ情報を共通するウィンドウ画面に表示することを特徴とする請求の範囲第５項に記載のプログラム開発支援装置。

７．　前記汎用言語はＣ言語であり、前記独自仕様言語ソースプログラムは、前記汎用言語ソースプログラムのプリプロセッサコマンドによって該汎用言語ソースプログラム内に記述されたことを特徴とする請求の範囲第１項に記載のプログラム開発支援装置。

８．　前記プリプロセッサコマンドは、＃ｐｒａｇｍａであることを特徴とする請求の範囲第７項に記載のプログラム開発支援装置。

９．　前記プログラム実行装置は、半導体試験装置であることを特徴とする請求項１に記載のプログラム開発支援装置。

１０．　汎用言語ソースプログラムのオブジェクトコードと独自仕様言語ソースプログラムのオブジェクトコードが混在したプログラムファイルを実行するプログラム実行装置において、

前記プログラムファイルの実行開始時に、前記汎用言語ソースプログラムのオブジェクトコードと前記独自仕様言語ソースプログラムのオブジェクトコードをメモリにロードすることを特徴とするプログラム実行装置。

１１．　前記プログラム実行装置は半導体試験装置であることを特徴とする請求の範囲第１０項に記載のプログラム実行装置。

１２．　汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置上で実行可能なプログラムファイルを作成するためのコンパイル方法において、

前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出する独自仕様言語ソースプログラム抽出ステップと、

抽出された独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイルステップと、

前記異種言語混在ソースプログラムのうちの前記汎用言語ソースプログラム記述部分をコンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイルステップと、

前記独自仕様言語オブジェクトコードと前記汎用言語オブジェクトコードを結合してオブジェクトファイルを生成するオブジェクトファイル生成ステップと、

前記オブジェクトファイル生成ステップによって生成された少なくとも一つの

オブジェクトファイルから前記プログラムファイルを生成するリンクステップと、

を含んだことを特徴とするコンパイル方法。

１３．　汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから作成された所定のプログラム実行装置上で実行可能なプログラムファイルをデバッグするためのデバッグ方法において、

前記異種言語混在ソースプログラム内のステートメントにブレークポイントを設定するブレークポイント設定ステップと、

前記プログラムファイルの実行時に前記ブレークポイントで該プログラムファイルを停止させるとともに、停止したプログラムファイルの前記ステートメントが前記汎用言語ソースプログラムに属する場合に汎用言語デバッガを起動し、停止したプログラムファイルの前記ステートメントが前記独自仕様言語ソースプログラムに属する場合に独自仕様言語デバッガを起動するデバッガ起動ステップと、

前記汎用言語デバッガと前記独自仕様言語デバッガとから得られたデバッグ情報を共通するウィンドウ画面に表示するデバッグ情報表示ステップと、

を含んだことを特徴とするデバッグ方法。

１４．　汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置上で実行可能なプログラムファイルを作成するためのステップをコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体において、

前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出する独自仕様言語ソースプログラム抽出ステップと、

抽出された独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイルステップと、

前記異種言語混在ソースプログラムのうちの前記汎用言語ソースプログラム記述部分をコンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイルステップと、

前記独自仕様言語オブジェクトコードと前記汎用言語オブジェクトコードを結合してオブジェクトファイルを生成するオブジェクトファイル生成ステップと、

前記オブジェクトファイル生成ステップによって生成された少なくとも一つのオブジェクトファイルから前記プログラムファイルを生成するリンクステップと、

を含むことを特徴とするコンピュータ読み取り可能な記録媒体。

１５．　汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから作成された所定のプログラム実行装置上で実行可能なプログラムファイルをデバッグするためのステップをコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体において、

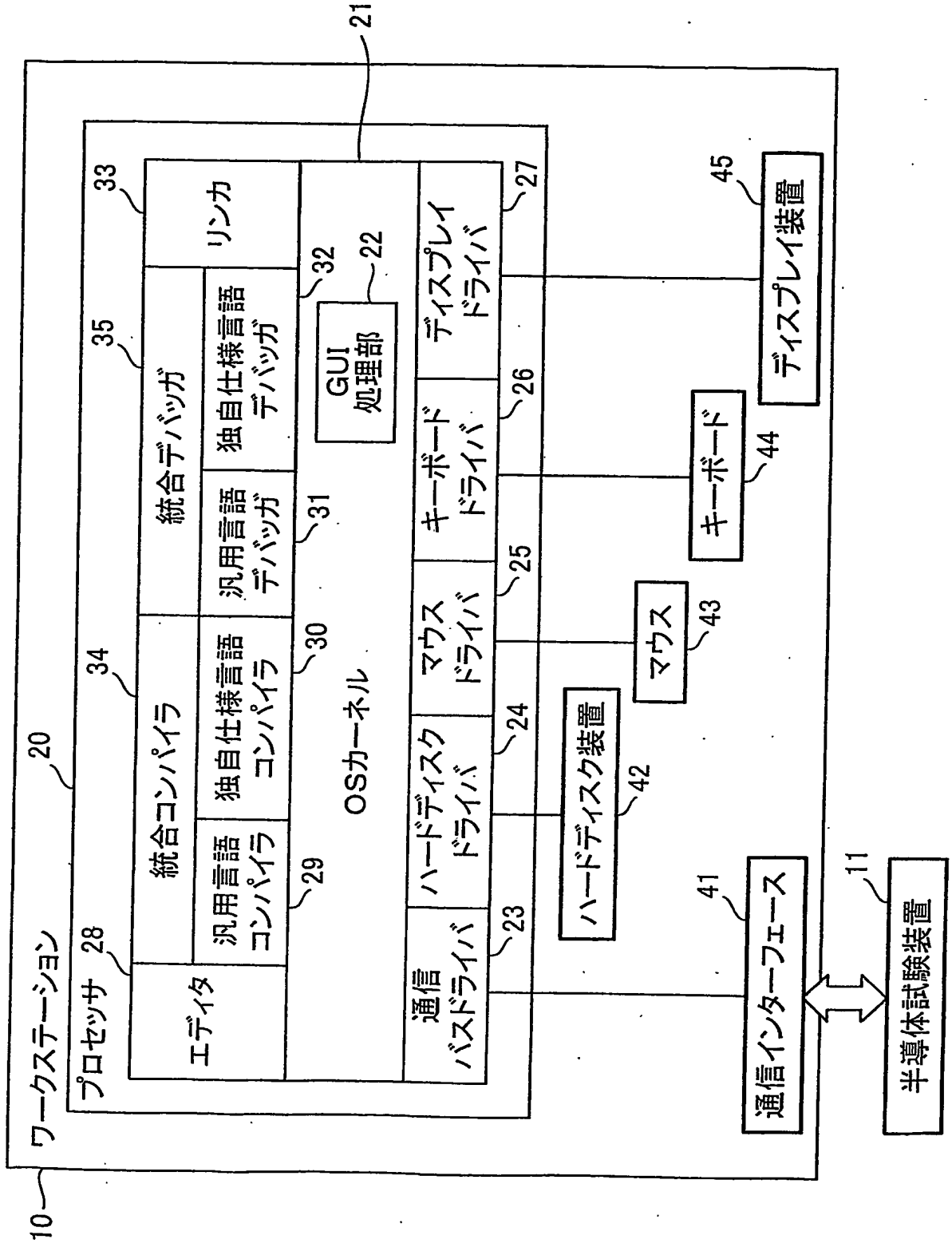
前記異種言語混在ソースプログラム内のステートメントにブレークポイントを設定するブレークポイント設定ステップと、

前記プログラムファイルの実行時に前記ブレークポイントで該プログラムファイルを停止させるとともに、停止したプログラムファイルの前記ステートメントが前記汎用言語ソースプログラムに属する場合に汎用言語デバッガを起動し、停止したプログラムファイルの前記ステートメントが前記独自仕様言語ソースプログラムに属する場合に独自仕様言語デバッガを起動するデバッガ起動ステップと、

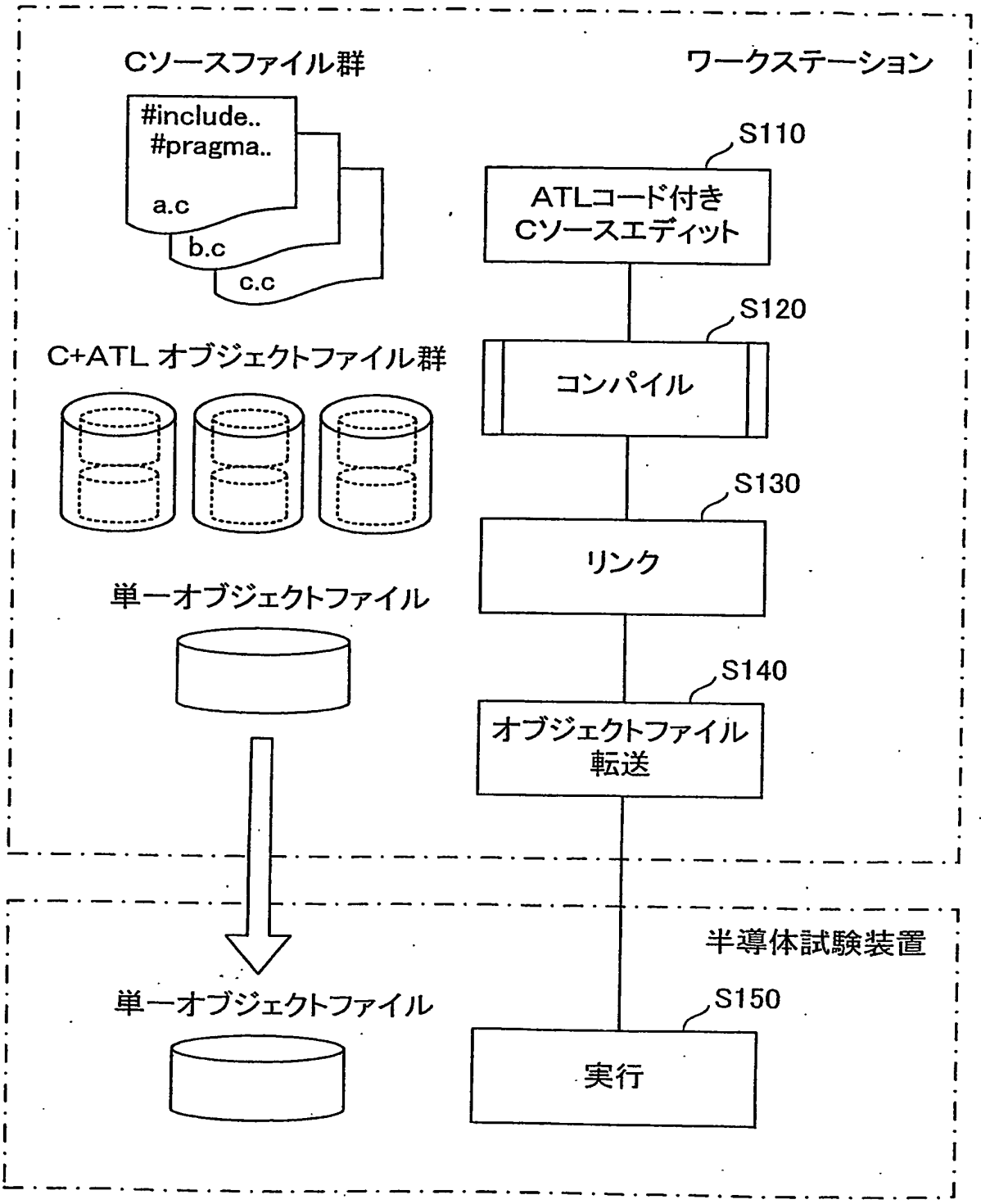
前記汎用言語デバッガと前記独自仕様言語デバッガとから得られたデバッグ情報を共通するウィンドウ画面に表示するデバッグ情報表示ステップと、

を含んだことを特徴とするコンピュータ読み取り可能な記録媒体。

第１図


10
ワークステーション
20
プロセッサ
28
エディタ
統合コンパイラ
34
汎用言語コンパイラ
29
独自仕様言語コンパイラ
30
統合デバッガ
35
汎用言語デバッガ
31
独自仕様言語デバッガ
32
リンカ
33
OSカーネル
21
GUI処理部
22
通信バスドライバ
23
ハードディスクドライバ
24
マウスドライバ
25
キーボードドライバ
26
ディスプレイドライバ
27
ハードディスク装置
42
マウス
43
キーボード
44
ディスプレイ装置
45
通信インターフェース
41
半導体試験装置
11

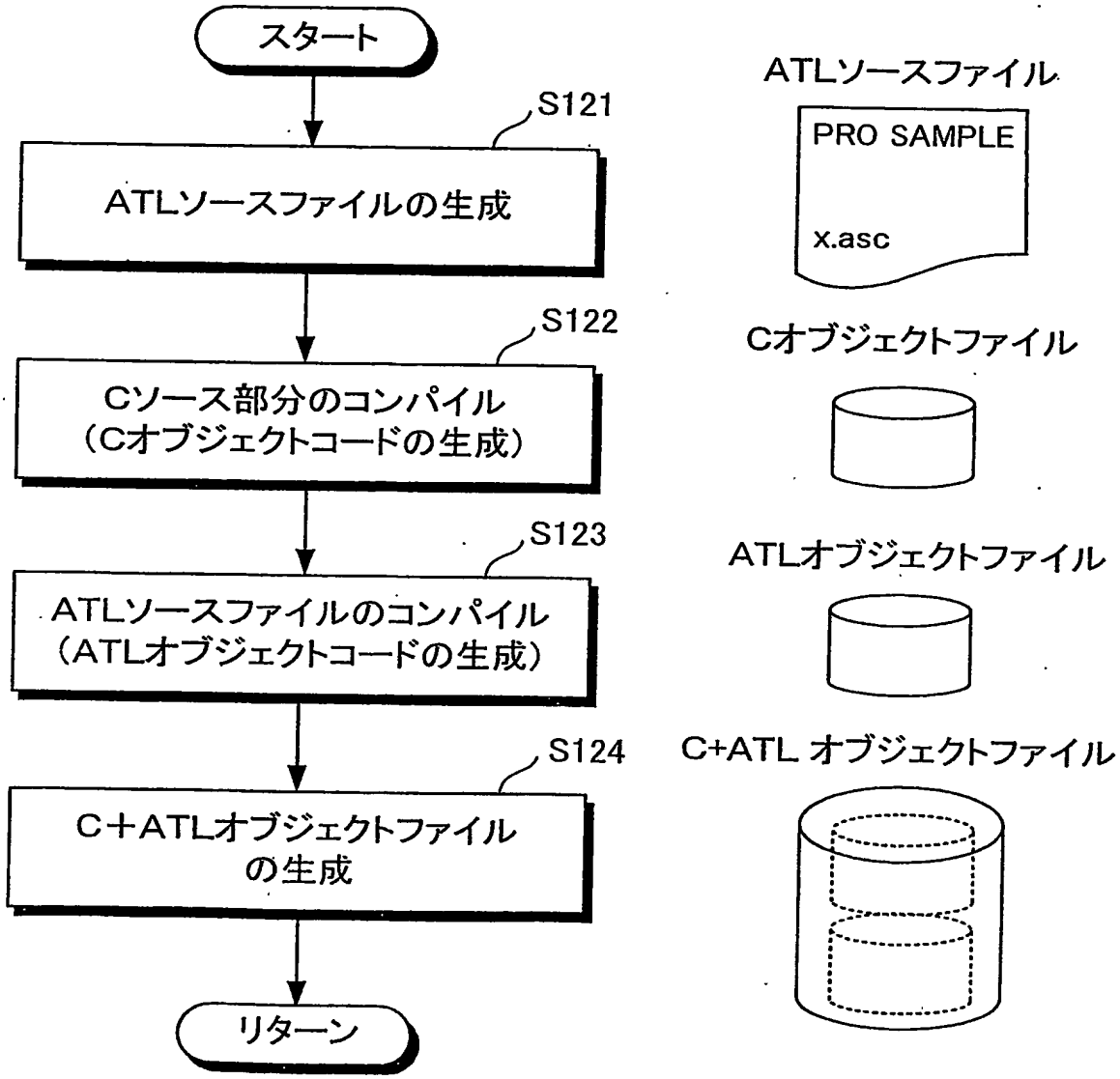
第２図


Cソースファイル群
#include..
#pragma..
a.c
b.c
c.c
C+ATL オブジェクトファイル群
単一オブジェクトファイル
単一オブジェクトファイル
ワークステーション
S110
ATLコード付き
Cソースエディット
S120
コンパイル
S130
リンク
S140
オブジェクトファイル
転送
半導体試験装置
S150
実行

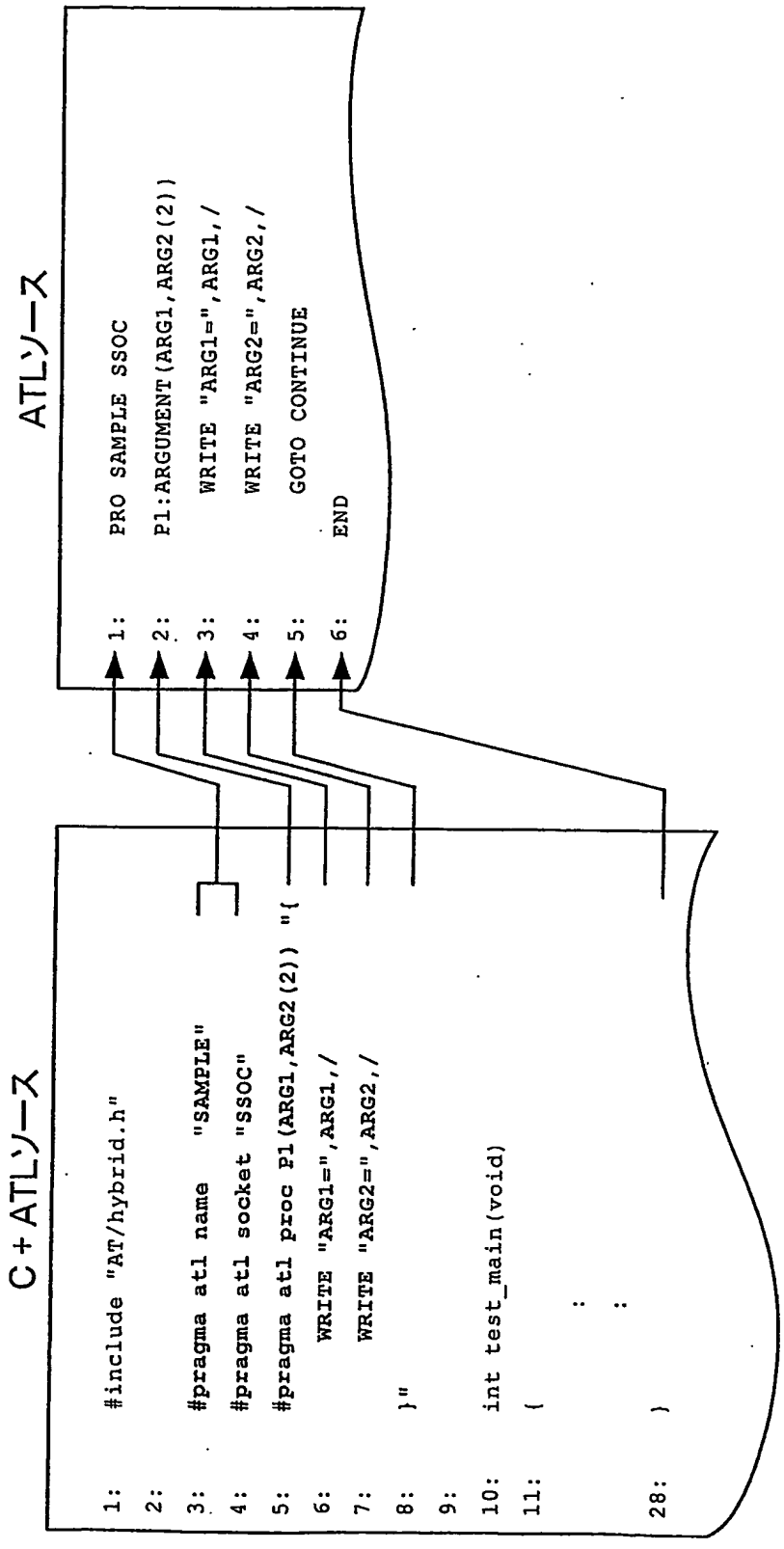
第３図

```
1:      #include "AT/hybrid.h"
2:
3:      #pragma atl name     "SAMPLE"
4:      #pragma atl socket   "SSOC"
5:      #pragma atl proc P1(ARG1,ARG2(2)) "{
6:           WRITE "ARG1=",ARG1,/
7:           WRITE "ARG2=",ARG2,/
8:      }"
9:
10:     int test_main(void)
11:     {
12:      int idata;
13:      struct atreal rdata[2];
14:
15:      if(ATL_init2(EM_ATL) != 0){
16:         AT_perror("ATL_init2()");
17:             exit(1);
18:      }
19:
20:      idata = 1;
21:      AT_strtorm(rdata, 2, "0.5V", "1.2V");
22:      if(ATL_proc(EM_ATL, "P1", INTE(&idata), DREAL(rdata, 2)) != 0){
23:          AT_perror("P2");
24:          exit(1);
25:      }
26:
27:      return(0);
28:      }
```

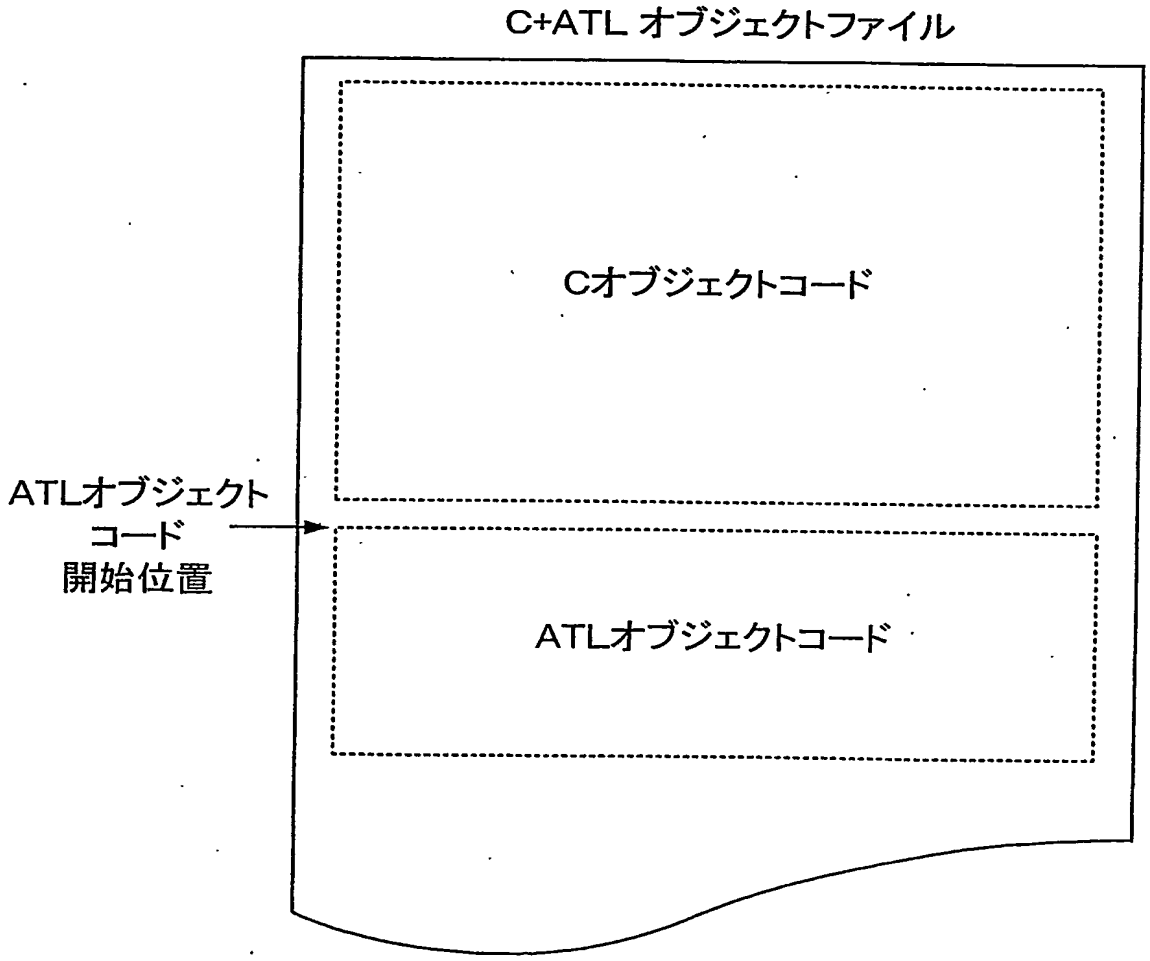
第４図


スタート
S121
ATLソースファイルの生成
S122
Cソース部分のコンパイル
（Cオブジェクトコードの生成）
S123
ATLソースファイルのコンパイル
（ATLオブジェクトコードの生成）
S124
C＋ATLオブジェクトファイル
の生成
リターン
ATLソースファイル
PRO SAMPLE
x.asc
Cオブジェクトファイル
ATLオブジェクトファイル
C+ATL オブジェクトファイル

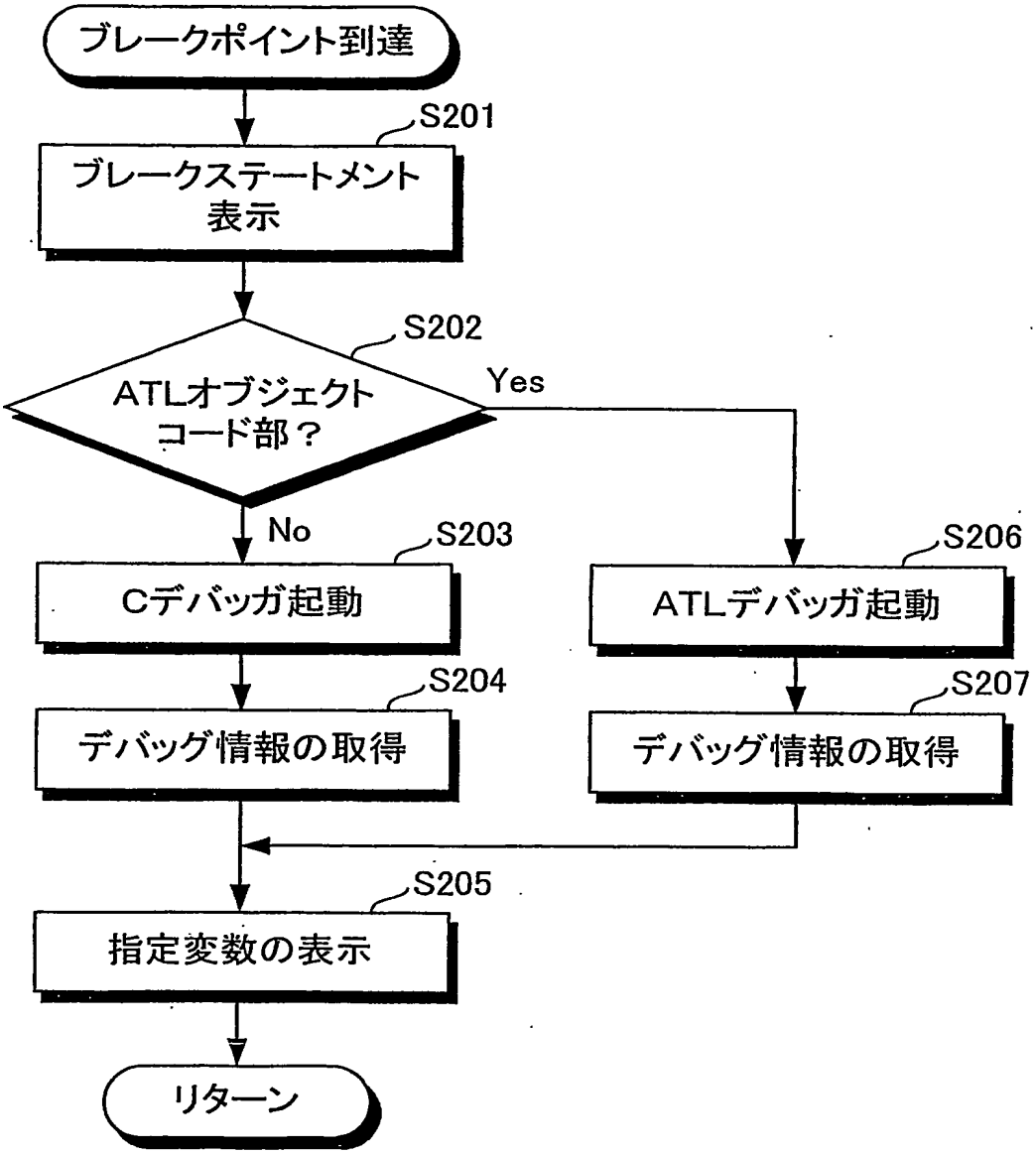
第５図


ATLソース
1: PRO SAMPLE SSOC
2: P1:ARGUMENT(ARG1,ARG2(2))
3: WRITE "ARG1=",ARG1,/
4: WRITE "ARG2=",ARG2,/
5: GOTO CONTINUE
6: END
C+ATLソース
1: #include "AT/hybrid.h"
2:
3: #pragma atl name "SAMPLE"
4: #pragma atl socket "SSOC"
5: #pragma atl proc P1(ARG1,ARG2(2)) "{
6: WRITE "ARG1=",ARG1,/
7: WRITE "ARG2=",ARG2,/
8: }"
9:
10: int test_main(void)
11: {
:
:
28: }

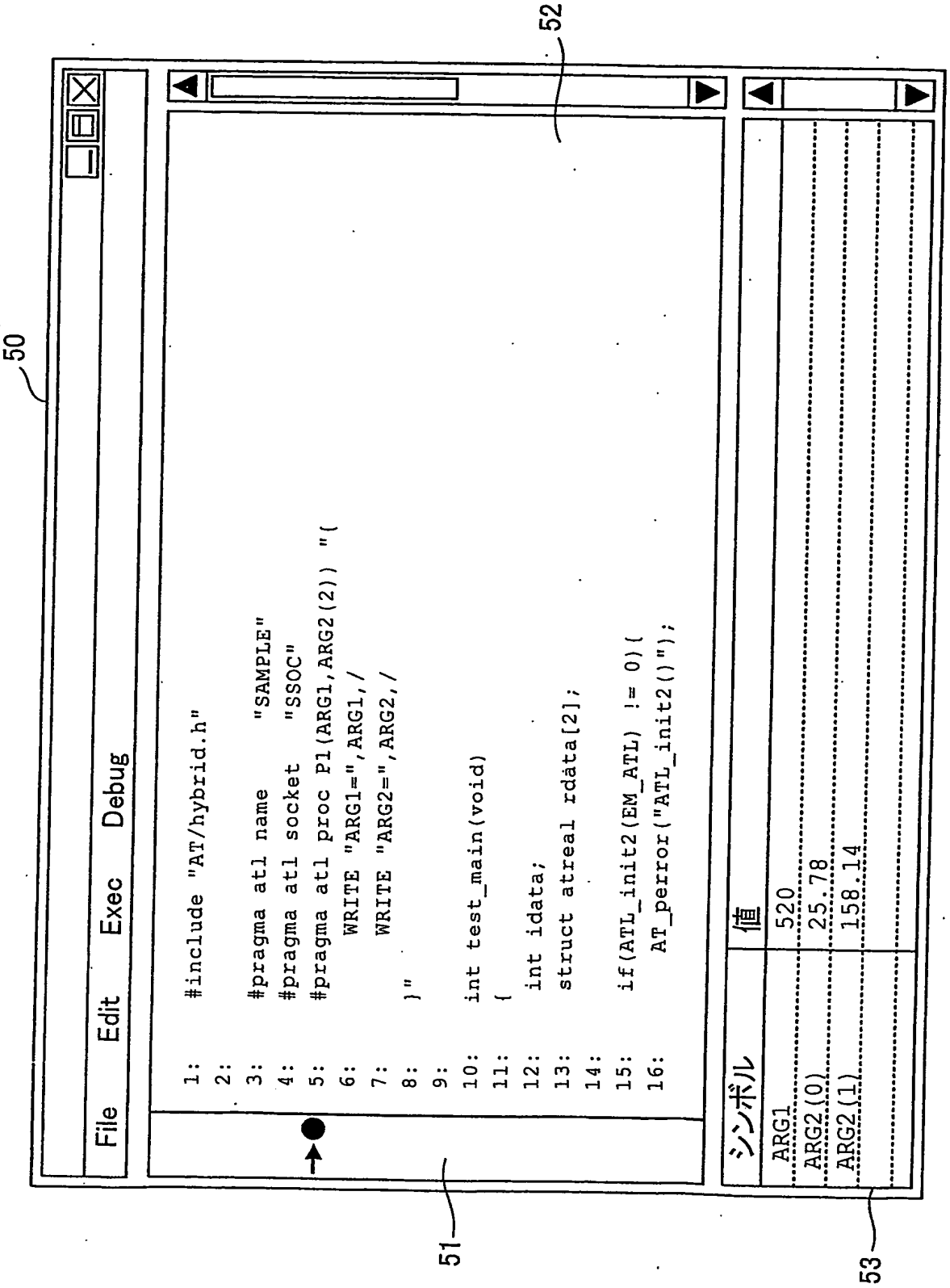
第６図


C+ATL オブジェクトファイル
Cオブジェクトコード
ATLオブジェクト
コード
開始位置
ATLオブジェクトコード

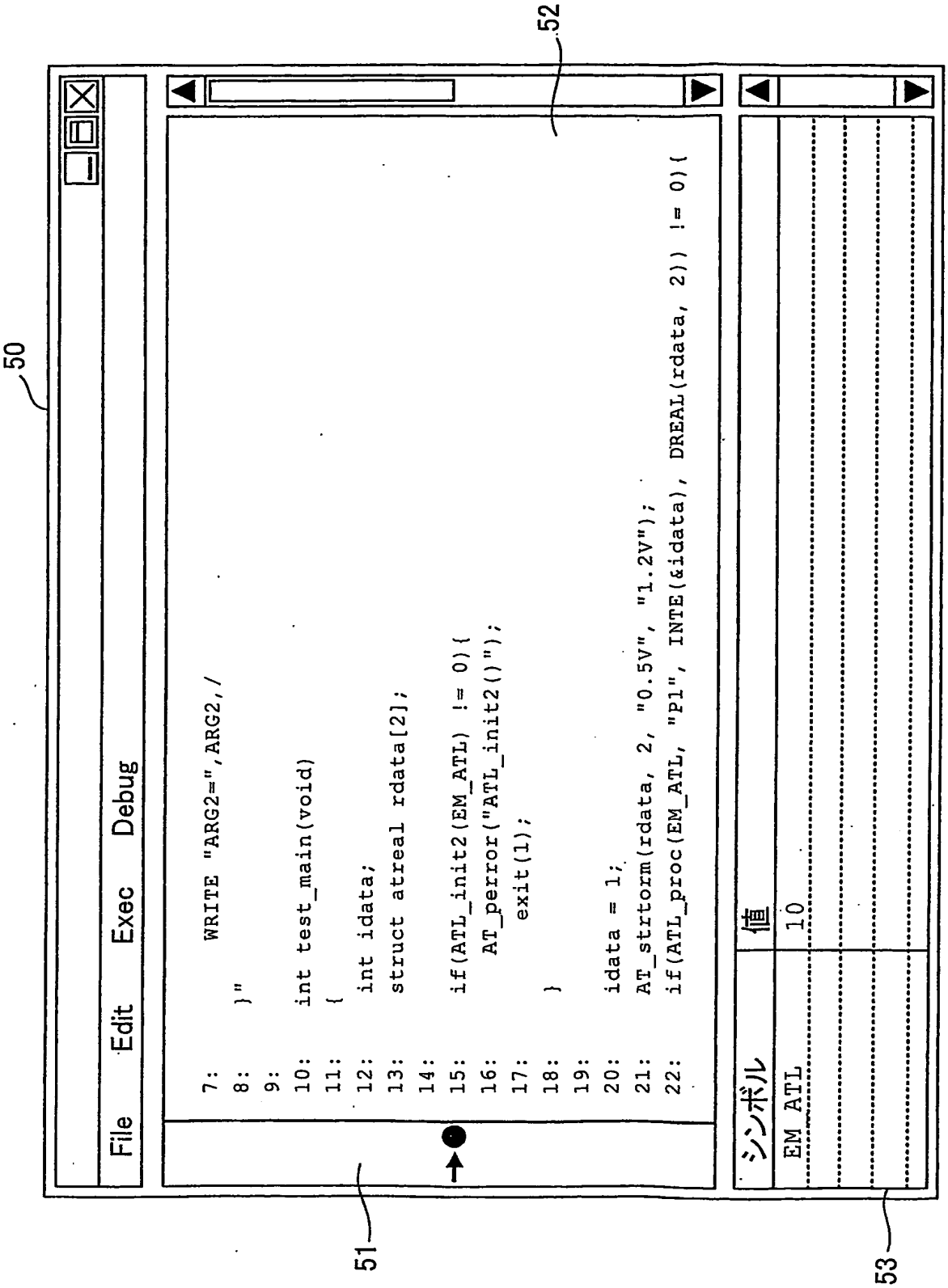
第7図


ブレークポイント到達
S201
ブレークステートメント表示
S202
ATLオブジェクトコード部？
Yes
No
S203
Cデバッガ起動
S206
ATLデバッガ起動
S204
デバッグ情報の取得
S207
デバッグ情報の取得
S205
指定変数の表示
リターン

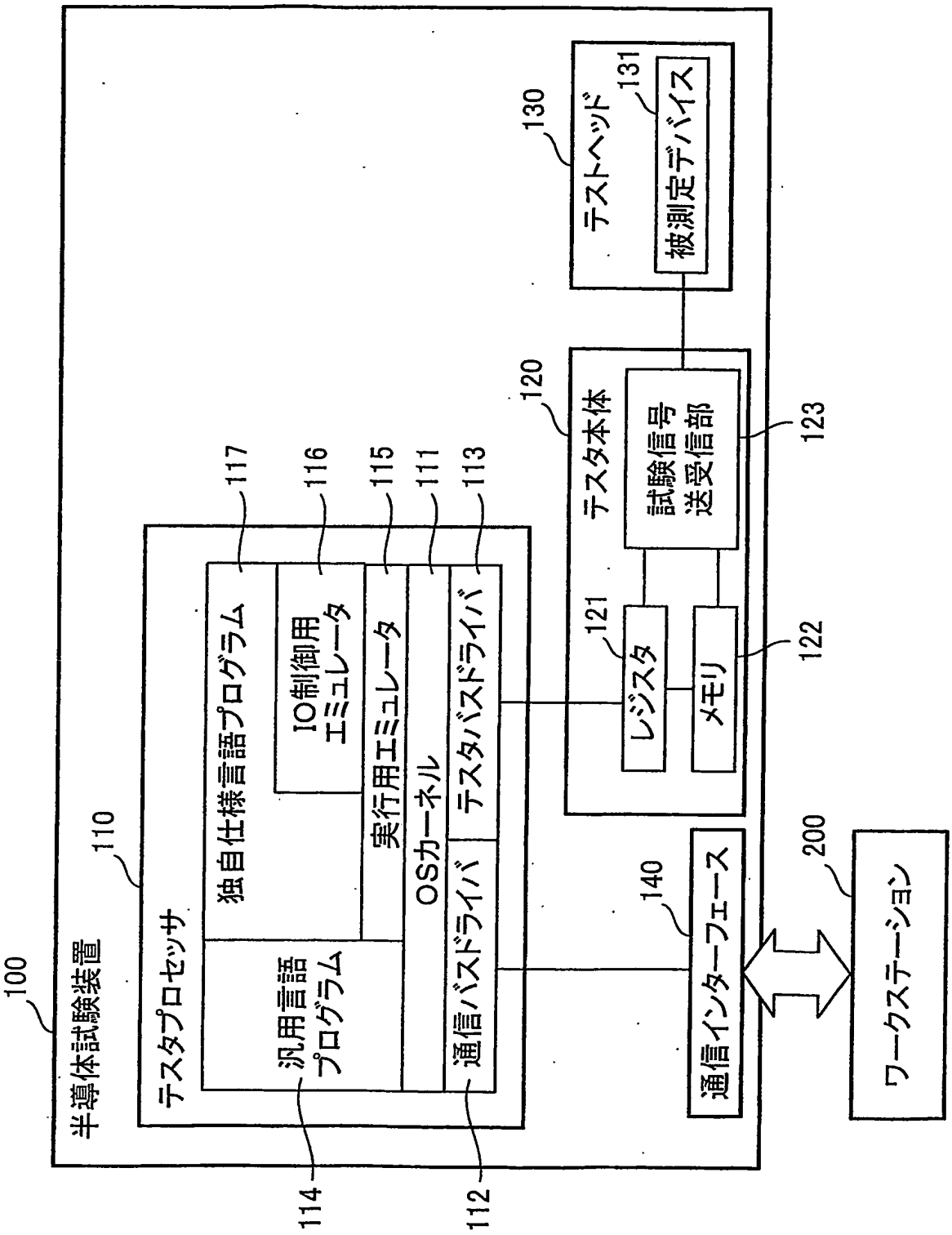
第８図


50
52
51
53
File Edit Exec Debug
1: #include "AT/hybrid.h"
2:
3: #pragma atl name "SAMPLE"
4: #pragma atl socket "SSOC"
5: #pragma atl proc P1(ARG1,ARG2(2)) "(
6: WRITE "ARG1=",ARG1,/
7: WRITE "ARG2=",ARG2,/
8:)"
9:
10: int test_main(void)
11: {
12: int idata;
13: struct atreal rdata[2];
14:
15: if(ATL_init2(EM_ATL) != 0){
16: AT_perror("ATL_init2()");
シンボル
値
ARG1
520
ARG2(0)
25.78
ARG2(1)
158.14

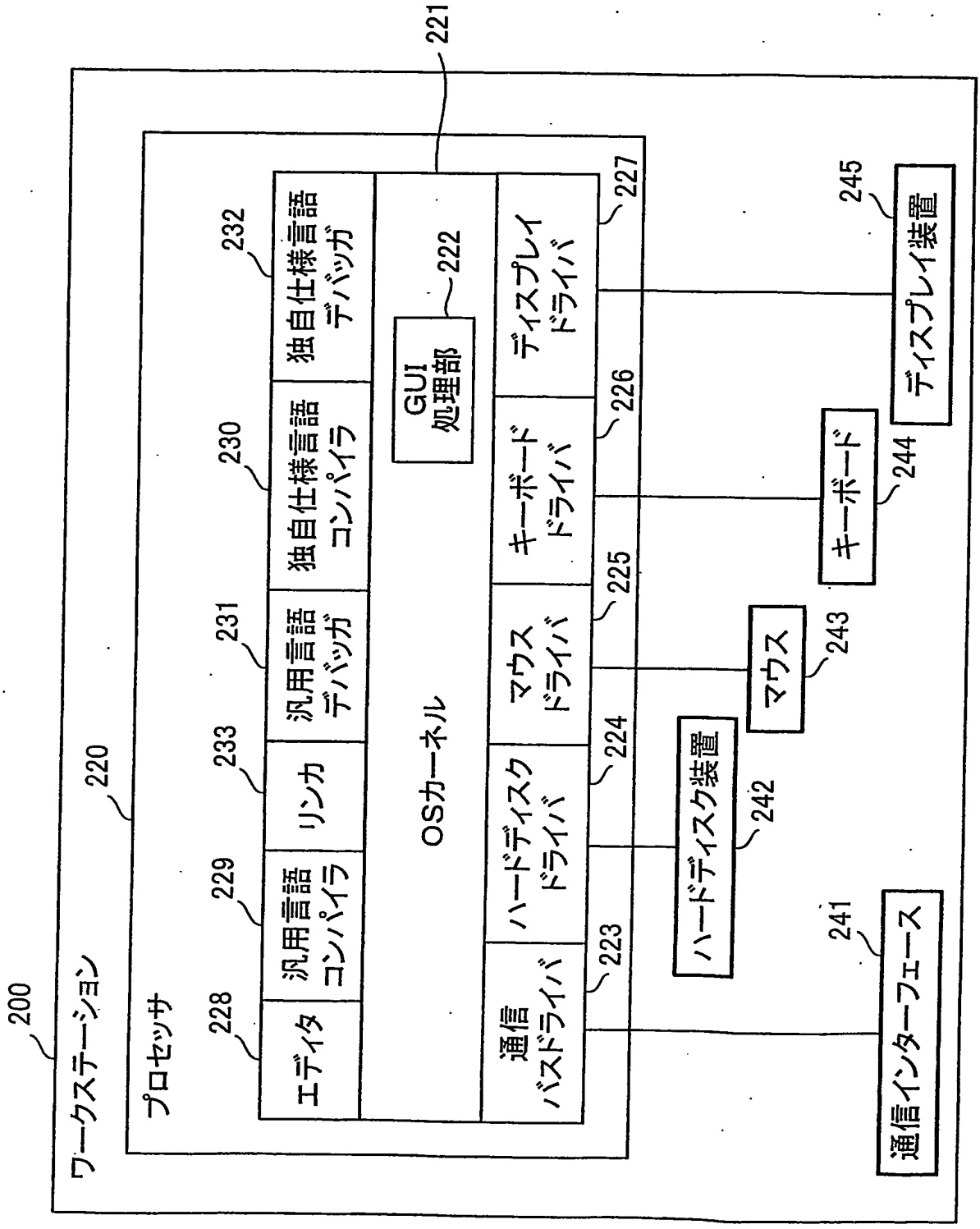
第９図


50
52
File Edit Exec Debug
7: WRITE "ARG2=",ARG2,/
8:)"
9:
10: int test_main(void)
11: {
12: int idata;
13: struct atreal rdata[2];
14:
15: if(ATL_init2(EM_ATL) != 0){
16: AT_perror("ATL_init2()");
17: exit(1);
18: }
19:
20: idata = 1;
21: AT_strtorm(rdata, 2, "0.5V", "1.2V");
22: if(ATL_proc(EM_ATL, "P1", INTE(&idata), DREAL(rdata, 2)) != 0){
シンボル
値
EM ATL
10
51
53

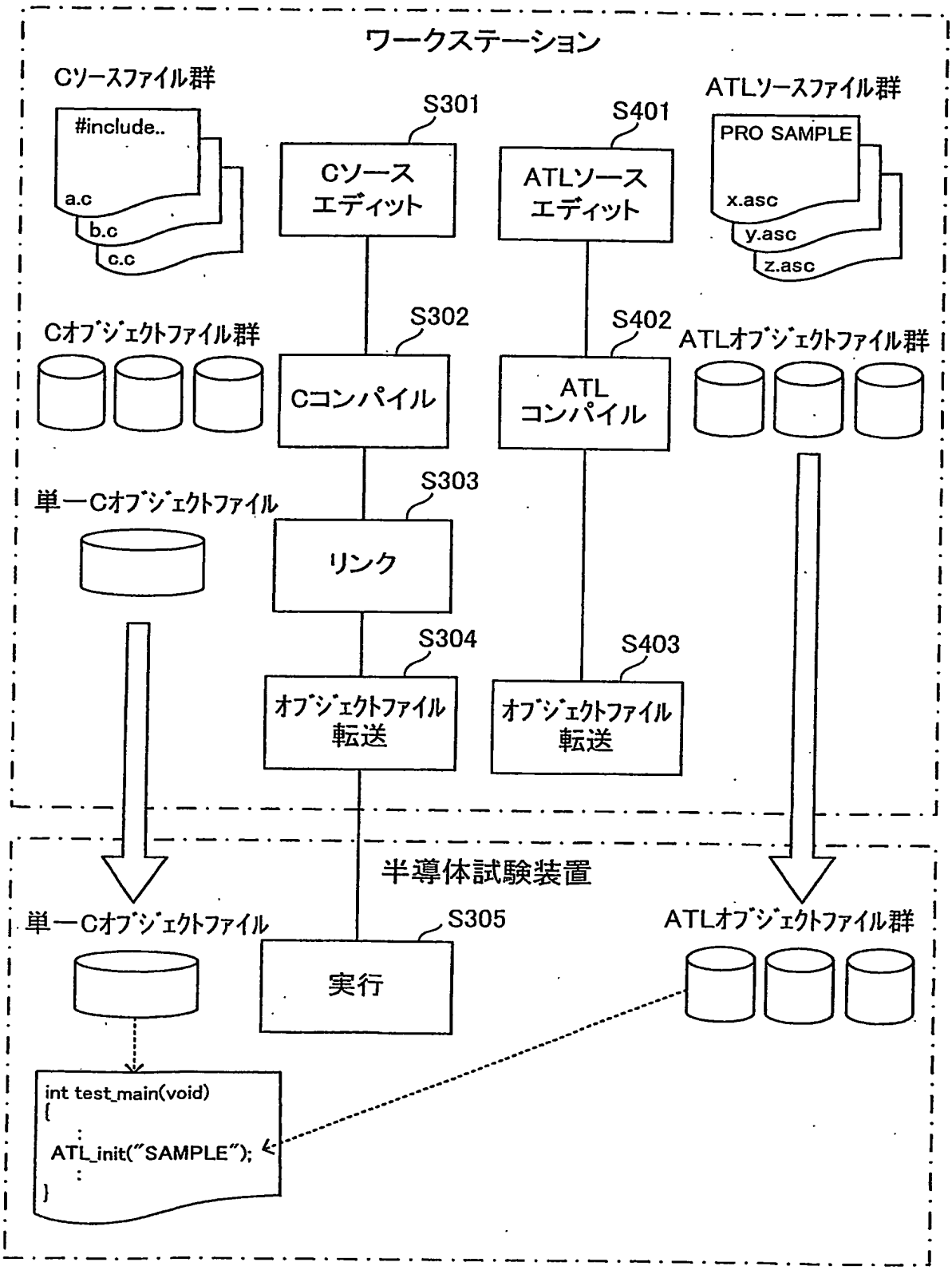
第１０図


100
半導体試験装置
110
テスタプロセッサ
117
独自仕様言語プログラム
116
IO制御用
エミュレータ
115
実行用エミュレータ
111
OSカーネル
113
テスタバスドライバ
汎用言語
プログラム
通信バスドライバ
114
112
130
テストヘッド
131
被測定デバイス
120
テスタ本体
試験信号
送受信部
123
121
レジスタ
メモリ
122
140
通信インターフェース
200
ワークステーション

第１１図


200
ワークステーション
220
プロセッサ
221
OSカーネル
222
GUI処理部
228
エディタ
229
汎用言語コンパイラ
233
リンカ
231
汎用言語デバッガ
230
独自仕様言語コンパイラ
232
独自仕様言語デバッガ
223
通信バスドライバ
224
ハードディスクドライバ
225
マウスドライバ
226
キーボードドライバ
227
ディスプレイドライバ
241
通信インターフェース
242
ハードディスク装置
243
マウス
244
キーボード
245
ディスプレイ装置

第１２図


ワークステーション
Cソースファイル群
#include..
a.c
b.c
c.c
S301
Cソース
エディット
S401
ATLソース
エディット
ATLソースファイル群
PRO SAMPLE
x.asc
y.asc
z.asc
S302
Cコンパイル
S402
ATL
コンパイル
Cオブジェクトファイル群
ATLオブジェクトファイル群
単一Cオブジェクトファイル
S303
リンク
S304
オブジェクトファイル
転送
S403
オブジェクトファイル
転送
半導体試験装置
単一Cオブジェクトファイル
S305
実行
ATLオブジェクトファイル群
int test_main(void)
{
ATL_init("SAMPLE");
}

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/13302

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F9/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Jitsuyo Shinan Koho 1922-1996 Toroku Jitsuyo Shinan Koho 1994-2003
Kokai Jitsuyo Shinan Koho 1971-2003 Jitsuyo Shinan Toroku Koho 1996-2003

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 11-212807 A (Hitachi, Ltd.), 06 August, 1999 (06.08.99), Full text; Figs. 1 to 4 (Family: none)	1-15
A	JP 7-319729 A (Hitachi, Ltd.), 08 December, 1995 (08.12.95), Full text; Figs. 1 to 5 (Family: none)	1-15
A	JP 2002-268896 A (Hitachi, Ltd.), 20 September, 2002 (20.09.02), Full text; Figs. 1 to 6 (Family: none)	1-15

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
04 December, 2003 (04.12.03)	16 December, 2003 (16.12.03)
Name and mailing address of the ISA/ Japanese Patent Office	Authorized officer
Facsimile No.	Telephone No.

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/13302

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2-146630 A (Fujitsu Maikon Systems Kabushiki Kaisha), 05 June, 1990 (05.06.90), Full text; Figs. 1 to 11 (Family: none)	1-15
A	JP 11-110256 A (Toshiba Corp.), 23 April, 1999 (23.04.99), Full text; Figs. 1 to 11 (Family: none)	1-15

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ０３／１３３０２

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Ｉｎｔ．Ｃｌ' Ｇ０６Ｆ９／４４

Ｂ．　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Ｉｎｔ．Ｃｌ' Ｇ０６Ｆ９／４４

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２００３年
日本国登録実用新案公報	１９９４－２００３年
日本国実用新案登録公報	１９９６－２００３年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

Ｃ．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ａ	ＪＰ　１１－２１２８０７　Ａ（株式会社日立製作所） １９９９．０８．０６，全文，第１－４図（ファミリーなし）	１－１５
Ａ	ＪＰ　７－３１９７２９　Ａ（株式会社日立製作所） １９９５．１２．０８，全文，第１－５図（ファミリーなし）	１－１５
Ａ	ＪＰ　２００２－２６８８９６　Ａ（株式会社日立製作所） ２００２．０９．２０，全文，第１－６図（ファミリーなし）	１－１５

[x] Ｃ欄の続きにも文献が列挙されている。　　[ ] パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日 ０４．１２．０３	国際調査報告の発送日 **16.12.03**
国際調査機関の名称及びあて先 日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	特許庁審査官（権限のある職員） 後藤　和茂　印　５Ｂ　９４６３ 電話番号　０３－３５８１－１１０１　内線　６９０７

様式ＰＣＴ／ＩＳＡ／２１０（第２ページ）（１９９８年７月）

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ０３／１３３０２

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ａ	ＪＰ　２－１４６６３０　Ａ（富士通マイコンシステムズ株式会社） １９９０．０６．０５，全文，第１－１１図（ファミリーなし）	１－１５
Ａ	ＪＰ　１１－１１０２５６　Ａ（株式会社東芝） １９９９．０４．２３，全文，第１－１１図（ファミリーなし）	１－１５